Альтернативный экзамен по дискретной математике. "Меандры"

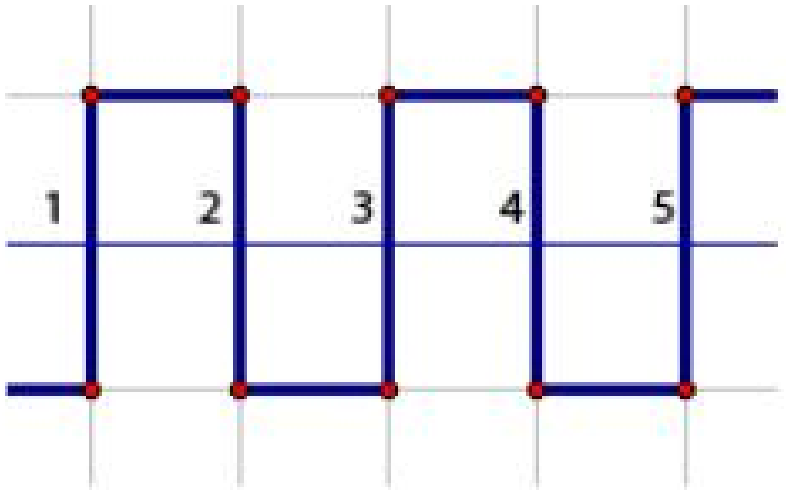
Выполнил: Варивода С. А.
Преподаватель: Поздняков С. Н.

Определение меандра

Меандр - это кривая без самопересечений, которая пресекает прямую несколько раз. Интуитивно, меандр можно расссмотреть как дорогу, пересекающую реку мостами в нескольких местах.

Меандр можно определить перестановкой, которая показывает в каком порядке река пересекает мосты.

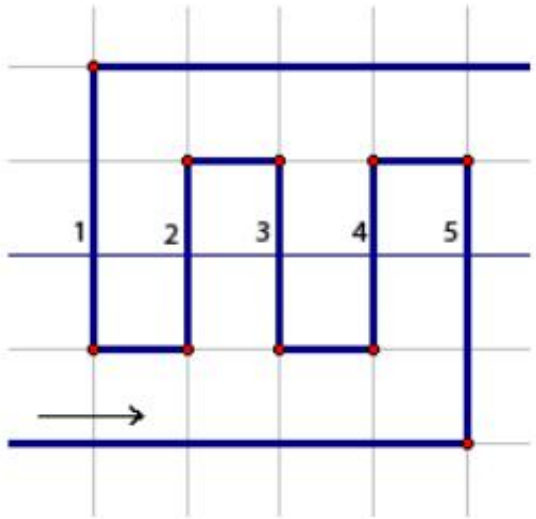
В нашем случае рассматривается не замкнутый меандр, который идет с запада на восток.



например, данный меандр определяется перестановкой 1,2,3,4,5

Постановка задач

- Первая задача: дана перестановка - определить, задаёт ли она меандр.
- Вторая задача: сгенерировать все меандры для N пересечений.



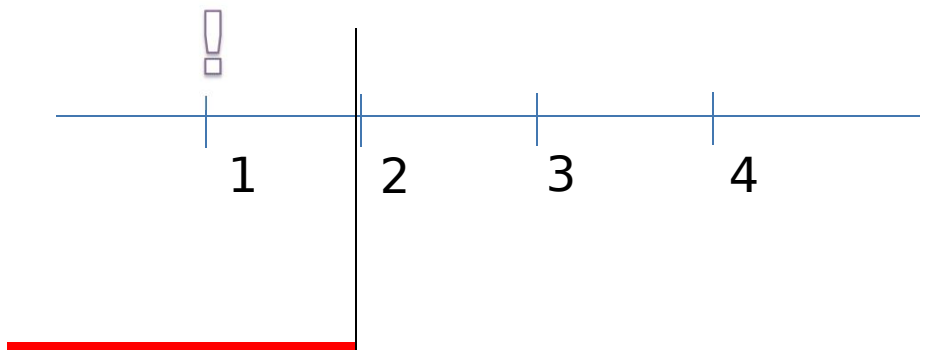
Решение первой задачи

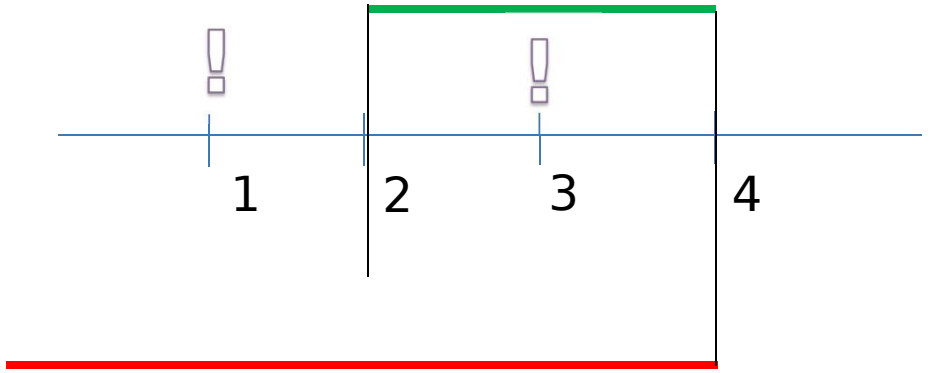
Проведя исследование, я заметил, что меандр должен соответствовать двум условиям:

1) Начинаться с нечетного моста
2) Дальше четные и нечетные мосты должны чередоваться

То есть четность позиции (если считать с 1) должна соответствовать четности номера моста.

Иначе возникают элементы, к которым "не подойти":





Решение первой задачи

Но оговоренных выше условий недостаточно, поэтому я предлагаю следующий алгоритм проверки меандра.

1) Создадим переменную-флаг, которая отмечает "снизу" или "сверху" мы пришли к i-ому элементу. Изначально флаг поставлен в положение "снизу".
2) Создадим два массива, верхних и нижних отрезков.
3) Проходя по перестановке циклом, совершаем следующее:

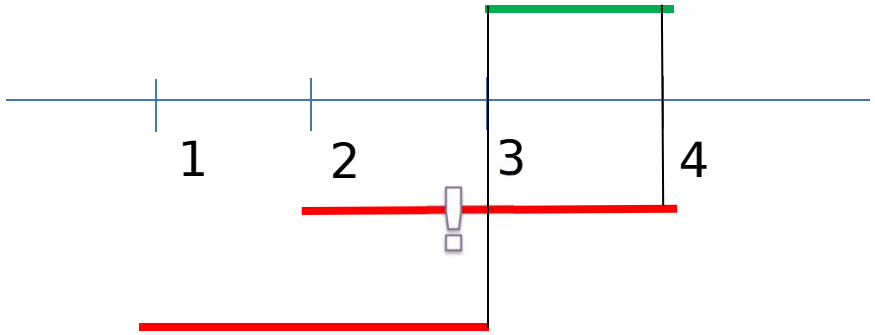
- Смотрим, попадает ли элемент в какой-либо из отрезков, в соответствующем для флага массиве ("снизу" - массив нижних отрезков, "сверху" - массив верхних отрезков).
- Если попадает, то проверяем предыдущий элемент, и если он не в отрезке, то это пересечение.
- Записываем в соответствующий массив (смотря на флаг) отрезок, левый элемент которого - элемент, от которого пришли, и правый - элемент, к которому пришли.

4) Если, дойдя до конца, мы не получили пересечение, то перестановка - меандр.

Решение первой задачи

Рассмотрим пример.

Перестановка: 3 4 2 1
Левая граница: 1
Флаг: снизу
Массив нижних отрезков (H): пустой
Массив верхних отрезков (B): пустой



Идем в цикле по перестановке.

1) Т.к. массивы пустые записываем в H (т.к. флаг снизу) [1, 3],
меняем флаг на "вверху" и левую границу на 3.

2) 4 не попадает в отрезки, поэтому просто добавляем в B [3,4],
меняем флаг на "снизу" и левую границу на 4.

3) К 2 мы пришли снизу и попадаем в отрезок содержащийся в H - [1,3].
Значит смотрим на предыдущий элемент. 4 вне этого отрезка.

Это не меандр.

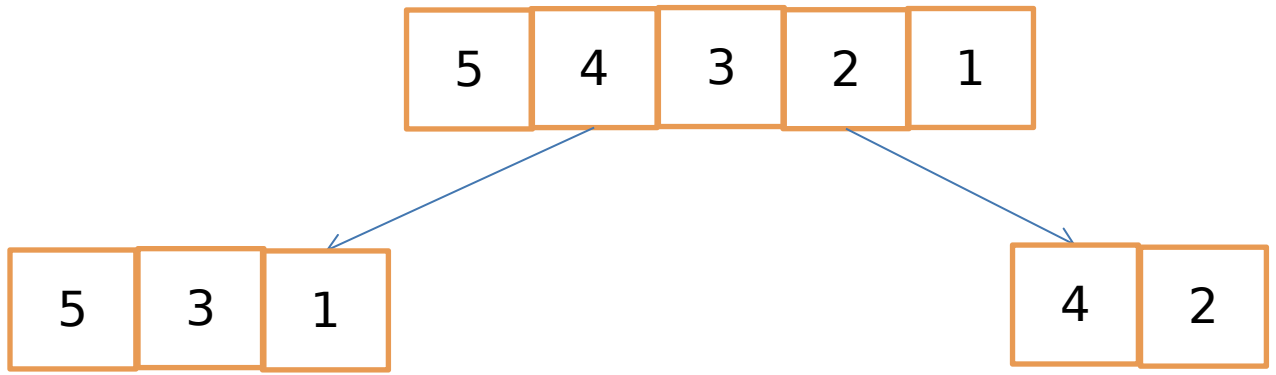
Решение второй задачи (1-ый алгоритм)

Воспользуемся стандартным алгоритмом генерации лексикографических перестановок без повторений. При получении новой перестановки будем проверять ее алгоритмом описанным выше.

Это позволит сгенерировать меандры для любого N.
Но из-за сложности алгоритма O(n!) при увеличении N время работы быстро увеличивается, поэтому такой алгоритм крайне неэффективен.

Решение второй задачи (2-ой алгоритм)

Чтобы ускорить 1-ый алгоритм я разделил перестановку на 2 массива четных и нечетных чисел.



И в двойном цикле мы делаем перестановки отдельно для каждого массива. После чего собираем обратно в один массив и проверяем на пересечения. Данный алгоритм будет генерировать сразу перестановки с чередующейся четностью элементов.

Сравнение алгоритмов

Оба алгоритма генерируют меандры для любого N. Но из-за сложности перебора может быть затрачено большое кол-во времени. Приведу стравнительную таблицу скорости нахождения всех меандров до от 6 до 15 (для меньших нет смысла проверять).

N	Алгоритм 1, миллисекунды	Алгоритм 2, миллисекунды
6	3	3
7	5	6
8	8	10
9	80	19
10	407	30
11	5121	100
12	61310	198
13	>120000	1448
14	>1000000	11948
15	>1000000	110179

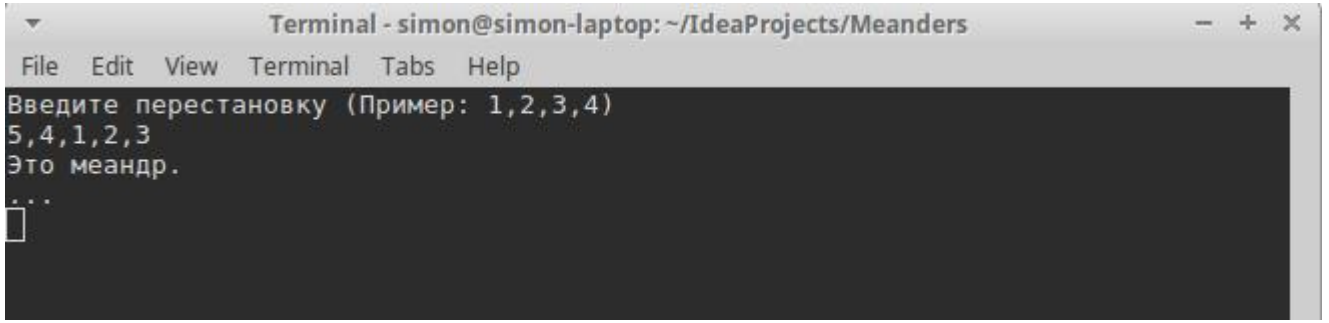
Сравнение полученных результатов

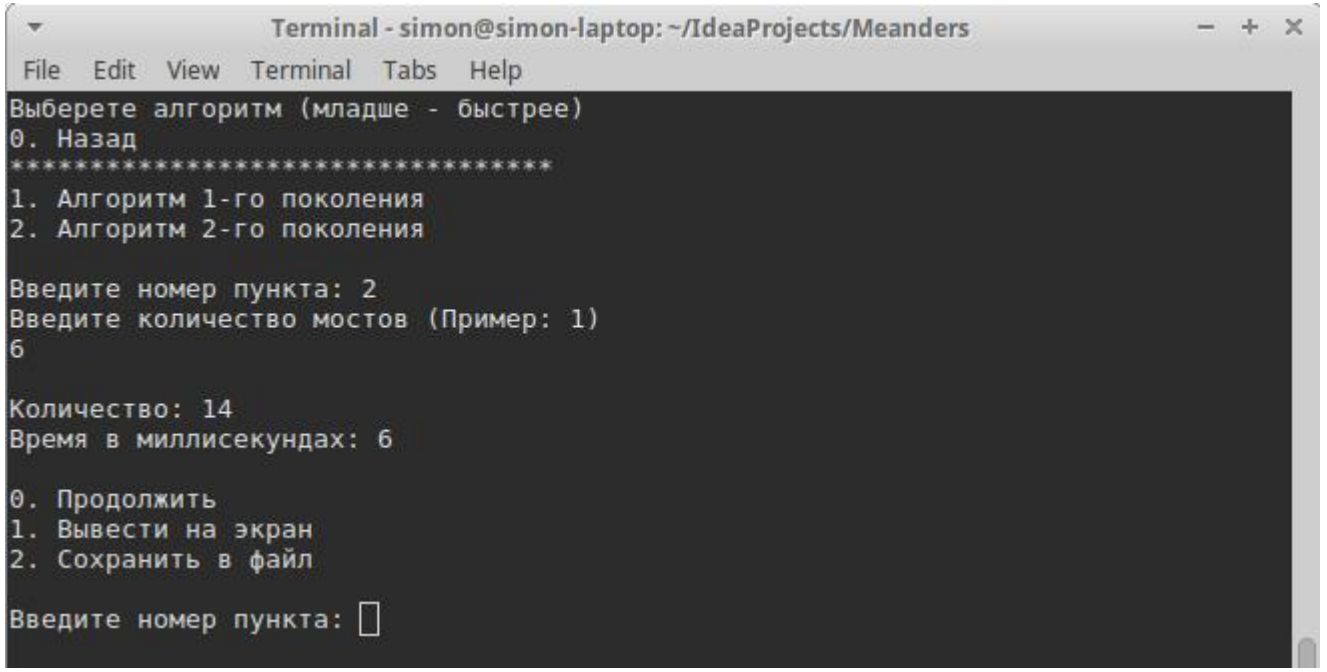
Сравнив известные меандры и полученные, мы можем убедиться, что алгоритмы работают верно.
Например, результаты полученные для 6:

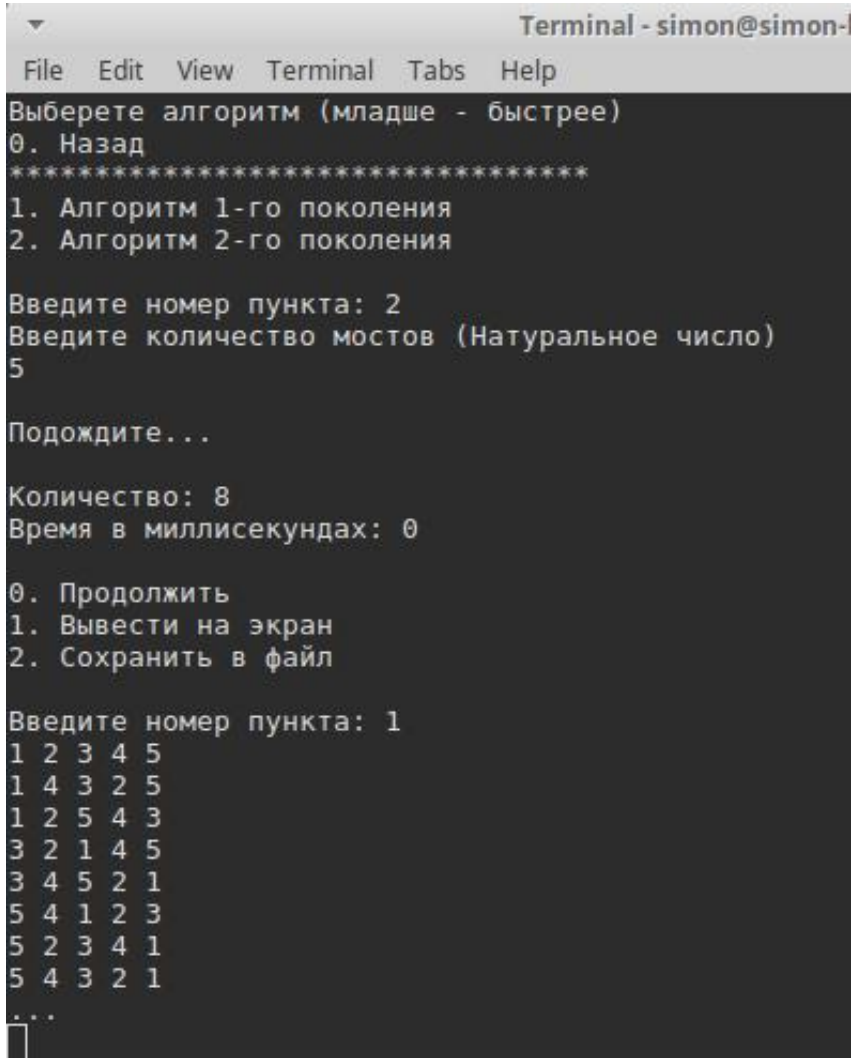
1 2 3 4 5 6
1 2 3 6 5 4
1 4 3 2 5 6
1 6 3 4 5 2
1 2 5 4 3 6
1 4 5 6 3 2
1 6 5 2 3 4
1 6 5 4 3 2
3 2 1 4 5 6
3 2 1 6 5 4
3 4 5 2 1 6
5 4 1 2 3 6
5 2 3 4 1 6
5 4 3 2 1 6

Эти результаты совпадают с теоретическими.
Все результаты генераций до 17 можно найти в репозитории.

Примеры работы программы







Используемые технологии

В качестве основного языка программирования выбран Java.

Выбор был сделан опираясь на главное преимущество этого языка - мультиплатформенность.